PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-271760
(43)Date of publication of application : 20.10.1995

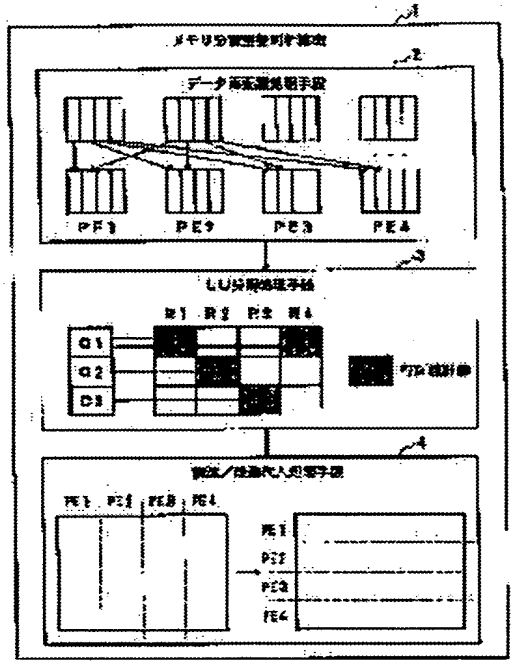
(51)Int.Cl. G06F 17/12
G06F 15/16
G06F 17/16

(21)Application number : 06-062241
(22)Date of filing : 31.03.1994

(71)Applicant : FUJITSU LTD
(72)Inventor : NAKANISHI MAKOTO

(54) METHOD AND COMPUTER FOR SIMULTANEOUS LINEAR EQUATION CALCULATING PROCESS BY MEMORY DECENTRALIZED TYPE PARALLEL COMPUTER

(57)Abstract:
PURPOSE: To dynamically perform optimum data arrangement for decentralizing a load, to shorten the transfer time of data, and to perform the transfer simultaneously with the calculation by dynamically rearranging an object matrix of LU decomposition, decentralized and arranged on respective processors, into arrangement by parallel transfer at a block level where column vectors are bundled.
CONSTITUTION: The data rearrangement processing means 2 of the memory decentralized type parallel computer 1 decentralizes and rearrange blocks of bundles of column vectors to the respective processors (PE). The data transfer required for the arrangement is performed in parallel and then speeded up. An LU decomposition processing means 3 transfers the time data for calculating a matrix product halfway in the LU decomposition of the blocks to the respective PEs. The time data are divided and transferred, and each PE performs calculation for the divided data; and this processing is repeated to perform the entire calculation. A forward/backward substitution processing means 4 rearranges the matrix into arrangement divided in a row vector direction and performs forward/backward substitution.



LEGAL STATUS

[Date of request for examination] 23.06.2000
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3639323
[Date of registration] 21.01.2005

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平7－271760

(43)公開日　平成７年(1995)10月20日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　17/12				
15/16	３９０　Ｚ			
17/16				
			Ｇ０６Ｆ　15/ 324	
			15/ 347	Ｋ

審査請求　未請求　請求項の数４　ＯＬ　（全 18 頁）

(21)出願番号　特願平6－62241

(22)出願日　平成６年(1994) ３月31日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地

(72)発明者　中西　誠
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

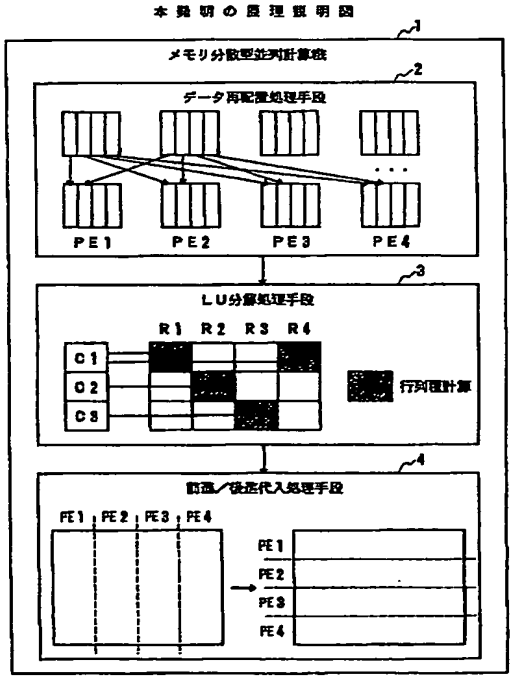
(74)代理人　弁理士　小笠原　吉義　（外２名）

(54)【発明の名称】　メモリ分散型並列計算機による連立１次方程式計算処理方法および計算機

(57)【要約】

【目的】ブロック化した外積型のＬＵ分解法により連立１次方程式を解くメモリ分散型並列計算機による連立１次方程式計算処理方法および計算機に関し，各プロセッサのＬＵ分解の負荷を均等にするとともに，実質的なデータ転送コストを削減し，並列性を高めて高速化を図ることを目的とする。

【構成】データを列ベクトルを束ねたブロックレベルでサイクリックな配置に並列転送で並べ換え，ＬＵ分解時に行列積の計算対象となるデータを分割し，分割したデータに対する行列積の計算と，データ転送とを並列に同時に実行し，ＬＵ分解した結果について，データを元の配置に戻し，さらに行列を行ベクトル方向に分割した配置になるように並べ換え，並列に前進／後進代入の処理を行う。



【特許請求の範囲】

【請求項１】　複数のプロセッサを備え，各プロセッサ間でデータ転送を行うことのできるメモリ分散型並列計算機(1) を用い，係数行列を各プロセッサに分配して，ブロック化した外積型のＬＵ分解法により連立１次方程式を解くメモリ分散型並列計算機による連立１次方程式計算処理方法において，前記メモリ分散型並列計算機の各プロセッサに分散されて配置されているＬＵ分解の対象となる行列を，列ベクトルを束ねたブロックレベルでサイクリックな配置に並列転送でもって動的に配置しなおす処理過程と，各プロセッサに配置されたブロックについてＬＵ分解を行う処理過程と，ＬＵ分解した結果について前進／後進代入処理を実行する処理過程とを有することを特徴とするメモリ分散型並列計算機による連立１次方程式計算処理方法。

【請求項２】　複数のプロセッサを備え，各プロセッサ間でデータ転送を行うことのできるメモリ分散型並列計算機(1) を用い，係数行列を各プロセッサに分配して，ブロック化した外積型のＬＵ分解法により連立１次方程式を解くメモリ分散型並列計算機による連立１次方程式計算処理方法において，各プロセッサに配置されたブロックについてＬＵ分解を行う途中で行列積の計算をするとき，行列積の計算対象となるデータを分割して各プロセッサに転送し，分割したデータに対する各プロセッサにおける行列積の計算と，他の分割したデータについての次の行列積の計算に用いる部分の並列転送とを同時に実行する処理を繰り返すことにより全体の計算を行う処理過程と，ＬＵ分解した結果について前進／後進代入処理を実行する処理過程とを有することを特徴とするメモリ分散型並列計算機による連立１次方程式計算処理方法。

【請求項３】　複数のプロセッサを備え，各プロセッサ間でデータ転送を行うことのできるメモリ分散型並列計算機(1) を用い，係数行列を各プロセッサに分配して，ブロック化した外積型のＬＵ分解法により連立１次方程式を解くメモリ分散型並列計算機による連立１次方程式計算処理方法において，前記メモリ分散型並列計算機の各プロセッサに分散されて配置されているＬＵ分解の対象となる行列を，列ベクトルを束ねたブロックレベルでサイクリックな配置に並列転送でもって動的に配置しなおす処理過程と，各プロセッサに配置されたブロックについてＬＵ分解を行う処理過程と，ＬＵ分解した結果について，ブロックレベルのサイクリックなデータ配置から行列を均等に列ベクトル方向に分割した配置を介して，行列を行ベクトル方向に分割した配置になるように各プロセッサ間でデータを並列に転送し，並列に前進／後進代入の処理を行う処理過程とを有することを特徴とするメモリ分散型並列計算機による連立１次方程式計算処理方法。

【請求項４】　複数のプロセッサを備え，各プロセッサ間でデータ転送を行うことのできるメモリ分散型並列計算機(1) であって，係数行列を各プロセッサに分配して，ブロック化した外積型のＬＵ分解法により連立１次方程式を解くメモリ分散型並列計算機において，ＬＵ分解の対象となる行列を，各プロセッサに対して，列ベクトルを束ねたブロックレベルでサイクリックな配置になるように並列転送により配置しなおすデータ再配置処理手段(2) と，ＬＵ分解における行列積の計算の際に，行列積の計算対象となるデータを分割して各プロセッサに転送し，分割したデータに対する各プロセッサにおける行列積の計算と，他の分割したデータについての次の行列積の計算に用いる部分の並列転送とを同時に実行する処理を繰り返すことにより全体の計算を行うＬＵ分解処理手段(3) と，ＬＵ分解した結果について，ブロックレベルのサイクリックなデータ配置から行列を均等に列ベクトル方向に分割した配置を介して，行列を行ベクトル方向に分割した配置に変え，並列に前進／後進代入の処理を行う前進／後進代入処理手段(4) とを備えたことを特徴とする連立１次方程式を解くメモリ分散型並列計算機。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は，複数のプロセッサ間で通信を行って処理を進めるマルチプロセッサシステムにより，高速に連立１次方程式を解くことができるようにしたメモリ分散型並列計算機による連立１次方程式計算処理方法および計算機に関する。

【０００２】連立１次方程式を高速に解く技術は，計算機の利用技術として非常に重要である。特に，高並列計算機により効率よく解く場合には，単なる数学的手法にとどまらず並列性を活かして，高並列計算機の特性を最大限に利用することのできる技術が必要になる。

【０００３】

【従来の技術】並列処理向きの連立１次方程式を解くアルゴリズムとして，ブロック化した外積型のＬＵ分解法が知られている。図２６は，そのブロック化した外積型のＬＵ分解法の概略を説明するための図である。

【０００４】外積形式のガウスの消去法をブロック化した方法で，図２６に示す配列Ａをｌｕ分解する。ブロック幅をｄとする。この方法では以下のような処理を行う。ｋ番目の処理で，更新部分$A^{(k)}$ を次の計算で更新する。

【０００５】

$$A^{(k)} = A^{(k)} - L2^{(k)} \cdot U2^{(k)} \quad \cdots\cdots(1)$$

ｋ＋１番目の処理では，$A^{(k)}$ をブロック幅ｄで分解して，ｄだけ小さいマトリックスを同じ式で更新する。

【０００６】$L2^{(k)}$，$U2^{(k)}$ は以下の式で求める必要がある。式(1) で更新を行う際に，

$$B^{(k)} = ((L1^{(k)})^T, (L2^{(k)})^T)^T U1^{(k)}$$

と分解し，
$U2^{(k)} = (L1^{(k)})^{-1} U2^{(k)}$
と更新する。
【０００７】このようなブロック化した外積型のＬＵ分解法をメモリ分散型並列計算機で実行する場合には，各プロセッサの負荷ができるだけ均等になるように，各プロセッサのメモリにデータを効率よく分配し，また各プロセッサ間で処理対象データの交換を効率よく行う必要がある。しかしながら，従来，ユーザインタフェースの簡略化などの面から，ブロック化されたデータを各プロセッサに順番に配置するようなことが考えられているだけであり，必ずしも各プロセッサのＬＵ分解の負荷が均等になってはいなかった。また，プロセッサ間でのデータ通信も並列性が十分でないため，通信コストの増大を招いていた。
【０００８】
【発明が解決しようとする課題】大規模な連立１次方程式を解くには，ＣＰＵの性能と大規模なメモリシステムが必要である。メモリ分散型のマルチプロセッサで連立１次方程式を高速に解くには，各プロセッサのメモリにデータを効率よく配置することと，効率のよいデータの転送を行う方式を考える必要がある。
【０００９】また，問題を解くユーザインタフェース（ホストのアプリケーションインタフェース）を煩雑にせずに実現する必要がある。本発明は上記問題点の解決を図り，性能を引き出すために負荷を分散する最適なデータ配置を動的に行い，データの転送時間が少なくかつ転送を計算と同時に行う方式を提供することを目的とする。
【００１０】
【課題を解決するための手段】図１は本発明の原理説明図である。本発明は，上記の問題を解決するため，連立１次方程式の解法の１つである外積形式のガウスの消去法をブロック化した方法を，以下のように実現する。
【００１１】メモリ分散型並列計算機１は，複数のプロセッサを備え，任意の２つのプロセッサが直接通信を行うことができる計算機である。
① メモリ分散型並列計算機１におけるデータ再配置処理手段２は，列ベクトルを束ねたブロックを各プロセッサ（以下，ＰＥという）に分散して配置しなおす。この配置に必要なデータ転送を，並列に行うことにより高速化する。
【００１２】② ＬＵ分解処理手段３は，ブロックのＬＵ分解を行う途中で行列積の計算を行うとき，データを各ＰＥに転送する。このときデータを分割して転送し，分割したデータに対する計算を各ＰＥで行い，これを繰り返すことにより全体の計算を行う。ここで，最初の転送時間が少なく，かつ以降の転送が計算と同時にできる方法を用い，実際の転送時間が計算時間と重なることにより非常に短くなったように見えるようにする。
【００１３】③ 前進／後進代入処理手段４は，ＬＵ分解したデータを前進／後進代入を行って並列に効率よく解くために，ブロックレベルのサイクリックなデータ配置から行列を均等に列ベクトル方向に分割した配置を介して，行列を行ベクトル方向に分割した配置に並べ換え，前進／後進代入を実行する。
【００１４】
【作用】ＬＵ分解を効率よく並列に実行するために，実際の計算を行う前に行列を列ベクトル方向に分割して配置していたものを，ブロックレベルでサイクリックな配置に動的に並列に配置しなおす。
【００１５】ＬＵ分解を行う上での行列積を効率よく並列に行うために，各プロセッサのベクトル処理と並列実行をバランスさせ，計算を行うのに必要な転送を，見かけ上，１つのプロセッサから１つのプロセッサへの転送に要する時間程度で行なえるようにする。
【００１６】ブロックレベルでサイクリックに並べ換えたデータ配置でＬＵ分解を行った後，ブロックレベルのサイクリックなデータ配置から行列を均等に列ベクトル方向に分割した配置を介して，行列を行ベクトル方向に分割した配置に変える。この転送を並列に行う。その結果について，並列に前進／後進代入の処理を行って解く。
【００１７】以上のように，データを並列転送で再配置することにより，ＬＵ分解の負荷を均等にし，１対ｎプロセッサ間通信のコストを見かけ上，１対１のプロセッサ間通信のコストに下げ，前進／後進代入処理における方程式を並列に解くことができるようにデータを並列転送で再配置する。
【００１８】
【実施例】以下，本発明の実施例を図を用いて説明する。図２は本発明の実施例に係るメモリ分散型並列計算機の例，図３は図２に示すプロセッサ（ＰＥ）の構成を示す図である。
【００１９】本発明は，例えば図２に示すようなハードウェアを持つメモリ分散型並列計算機によって実現される。各プロセッサ（ＰＥ）１０はクロスバーネットワーク１５に接続され，それぞれスカラ演算を行うスカラユニット１１と，ベクトル演算を行うベクトルユニット１２と，プログラムの命令列および演算対象データを記憶する主記憶装置１３と，任意の他のプロセッサとの間でクロスバーネットワーク１５を介して通信を行うＰＥ間通信ユニット１４とからなる。
【００２０】各プロセッサ１０は，例えば図３に示すように構成され，スカラユニット１１は，主記憶装置１３のデータを一時的に保持するキャッシュメモリ２１，演算に用いる汎用レジスタ／浮動小数点レジスタ２２，スカラ命令を実行するスカラ演算機２３などからなる。主記憶装置１３からフェッチした命令がベクトル命令であるときには，ベクトルユニット１２が起動される。ベク

トルユニット１２は，主記憶装置１３からデータをロードするためのロードパイプライン２４，主記憶装置１３へデータをストアするためのストアパイプライン２５，ベクトル演算対象の一連のデータを保持するベクトルレジスタ２６，特定の演算対象データをマスクするマスクレジスタ２７，演算対象データを指定するマスクパイプライン２８，ベクトルデータの乗算を行う乗算パイプライン２９，ベクトルデータの加減算または論理演算を実行する加算／論理演算パイプライン３０，ベクトルデータの除算を行う除算パイプライン３１を備える。

【００２１】次に，本発明により連立１次方程式を解く方式について詳細に説明する。

〔１〕動的にデータを並べ換える方法

初めに，動的にデータを並べ換える方法について説明する。

【００２２】図２に示すようなメモリ分散型並列計算機において，データは分散されて配置されている。二次元配列の場合，列方向の部分に分割して各プロセッサ（ＰＥ）１０に割り当てられる。この二次元配列の行列をある幅を持ったブロックを集めたものと考えて，このブロックを並べ換える。

【００２３】Ａ（ｎ，ｍ）＝Ａ（ｎ，１：ｄ）＋Ａ（ｎ，ｄ＋１：２ｄ）＋Ａ（ｎ，２ｄ＋１：３ｄ）＋…

＝Ａ１＋Ａ２＋Ａ３＋　…　＋Ａｋ

ただし，Ａｊ＝Ａ（ｎ，（ｊ－１）＊ｄ＋１：ｊ＊ｄ）

これを以下のように並べ換える。プロセッサ数を＃ｐｅとする（プロセッサｉ（ｉ＝１，…，＃ｐｅ））。

【００２４】ブロックＡｊをｍｏｄ（ｊ－１，＃ｐｅ）＋１となるプロセッサに割り付ける。ｍｏｄ（ａ，ｂ）は整数ａを整数ｂで割ったときの剰余を表す。配列Ａと同じ大きさの配列Ｂを，同じように各プロセッサに分散して割り付ける。図２に示すメモリ分散型並列計算機では，各プロセッサはクロスバーネットワーク１５に結合されていて，同時に転送を行うことができる。また，同じプロセッサに対して同時に読み込みと書き込みができる。この機能を使って上記の並べ換えを図４に示すような手順で行う。

【００２５】図４は本発明の実施例における並べ換えの処理フローを示す。各プロセッサ（ＰＥ）に，ブロックＡｊを順番に同じ数だけ並べてＡとする。ブロックの総数ｊｔはプロセッサ数＃ｐｅで割り切れるように並べる。１つのプロセッサにあるブロックの数＃ｂを，＃ｂ＝ｊｔ／＃ｐｅとする。

【００２６】並べ換えの処理では，まず図４に示すステップＳ１において，ｃｏｕｎｔ＝０とし，ｍｏｄ（ｎ１＊＃ｂ，＃ｐｅ）＝０である最小の正の整数ｎ１を探す。次に，ステップＳ２において，各ＰＥのＰＥ番号をｐｎｏ（ｐｎｏ＝１，…，＃ｐｅ）としたとき，各ＰＥでｋ＝１＋（ｐｎｏ－１）／ｎ１とする。

【００２７】ステップＳ３において，各ＰＥで次の計算をする。

ｐ＝（ｐｎｏ－１）＊＃ｂ＋ｋ

ｐ１＝（ｐ－１）／＃ｐｅ

ｐ２＝ｍｏｄ（ｐ－１，＃ｐｅ）

ｑ＝ｐ２＊＃ｂ＋ｐ１＋１

ステップＳ４において，Ｂｑ＝Ａｐの転送を各ＰＥで行う。ここでＡｐは各ＰＥにあり，Ｂｑは各々異なったＰＥになるため，転送は完全に並列に行うことができる。

【００２８】ステップＳ５において，ｃｏｕｎｔ＝ｃｏｕｎｔ＋１とする。ステップＳ６において，ｃｏｕｎｔ＞＃ｂであるかどうかを判定する。ｃｏｕｎｔ＞＃ｂであれば，この処理を終了し，ｃｏｕｎｔ＞＃ｂでなければステップＳ７の処理を行う。

【００２９】ステップＳ７において，各ＰＥで次の計算をする。

ｋ＝ｋ＋１

ｋ＝ｍｏｄ（ｋ－１，＃ｂ）＋１

この後，ステップＳ３へ戻り，同様に処理を繰り返す。

【００３０】図５は実施例におけるブロックの転送例を示す。図５の例では，＃ｐｅ＝４，＃ｂ＝４である。Ａ，Ｂにおける１つの矩形は１ブロックを表し，各ブロック内の数字は説明のためのブロックの番号を表す。図４の処理フローに示すように，ｍｏｄ（＃ｂ，＃ｐｅ）＝０のため，ｎ１＝１となり，ＰＥ１においてｋ＝１，ＰＥ２においてｋ＝２，ＰＥ３においてｋ＝３，ＰＥ４においてｋ＝４となる。したがって，１回目の転送パスでは，ＰＥ１の配列Ａの１番目のブロック１はＰＥ１の配列Ｂの１番目に，ＰＥ２の配列Ａの２番目のブロック６はＰＥ２の配列Ｂの２番目に，ＰＥ３の配列Ａの３番目のブロック１１はＰＥ３の配列Ｂの３番目に，ＰＥ４の配列Ａの４番目のブロック１６はＰＥ４の配列Ｂの４番目に転送される。

【００３１】続いて，２回目の転送パスにおいて，ｋ＝２（ＰＥ１），ｋ＝３（ＰＥ２），ｋ＝４（ＰＥ３），ｋ＝１（ＰＥ４）となり，ＰＥ１の配列Ａの２番目のブロック２はＰＥ２の配列Ｂの１番目に，ＰＥ２の配列Ａの３番目のブロック７はＰＥ３の配列Ｂの２番目に，ＰＥ３の配列Ａの４番目のブロック１２はＰＥ４の配列Ｂの３番目に，ＰＥ４の配列Ａの１番目のブロック１３はＰＥ１の配列Ｂの４番目に転送される。

【００３２】同様に，３回目の転送パスにおいて，ｋ＝３（ＰＥ１），ｋ＝４（ＰＥ２），ｋ＝１（ＰＥ３），ｋ＝２（ＰＥ４）となり，ＰＥ１の配列Ａの３番目のブロック３はＰＥ３の配列Ｂの１番目に，ＰＥ２の配列Ａの４番目のブロック８はＰＥ４の配列Ｂの２番目に，ＰＥ３の配列Ａの１番目のブロック９はＰＥ１の配列Ｂの３番目に，ＰＥ４の配列Ａの２番目のブロック１４はＰＥ２の配列Ｂの４番目に転送される。

【００３３】同様に，４回目の転送パスにおいて，ｋ＝

４（ＰＥ１），ｋ＝１（ＰＥ２），ｋ＝２（ＰＥ３），ｋ＝３（ＰＥ４）となり，ＰＥ１の配列Ａの４番目のブロック４はＰＥ４の配列Ｂの１番目に，ＰＥ２の配列Ａの１番目のブロック５はＰＥ１の配列Ｂの２番目に，ＰＥ３の配列Ａの２番目のブロック１０はＰＥ２の配列Ｂの３番目に，ＰＥ４の配列Ａの３番目のブロック１５はＰＥ３の配列Ｂの４番目に転送される。

【００３４】以上のように転送してデータを並べ換えることにより，１つのＰＥにおいて，同時に複数の読み込みまたは同時に複数の書き込みが起きるような衝突がなく，かつ同じＰＥでは同時に１つの読み込みと１つの書き込みができるので，転送は完全に並列に行われるとともに，衝突による待ち合わせが生じることはない。

【００３５】〔２〕行列積の効率的な方法

前述のようにして並べ換えたものについて，ブロック化したＬＵ分解を行う方法を以下に説明する。図６は，実施例におけるＬＵ分解の対象となる行列の例を示す。図２６で説明したように外積形式のガウスの消去法をブロック化した方法でＬＵ分解を実行する。そのため，図６に示す更新部分Ｕについて，Ｕ＝Ｕ－Ｃ×Ｒの計算を行って，Ｕを更新する。

【００３６】この計算では，行列Ｃを各ＰＥに転送する必要がある。考え方として簡単な方法は，単に行列Ｃ全体を各ＰＥに転送する方法である。行列Ｃの部分を各ＰＥに転送してから行列積の計算を行う場合，行列Ｃ全体を２分木の方法により各ＰＥに２の巾乗のパターンで転送を行う方法が考えられる。すなわち，ＰＥ１から残りの＃ｐｅ－１個のＰＥに転送することを考えた場合，次のように転送する。

【００３７】①ＰＥ１からＰＥ２へ行列Ｃを転送する。②次に，ＰＥ１からＰＥ３への転送と，ＰＥ２からＰＥ４への転送を同時に行う。③次に，ＰＥ１からＰＥ５へ，ＰＥ２からＰＥ６へ，ＰＥ３からＰＥ７へ，ＰＥ４からＰＥ８への転送を同時に行う。このような転送を続けると，全体の転送コストはＬＯＧ２（＃ｐｅ）のオーダとなる。

【００３８】本実施例では，この全体の転送コストを削減するために，行列Ｃを行方向に分割して計算を行う方法を採用する。以下にその方法を説明する。ブロック化したＬＵ分解のｋ番目のステージで，ＡｋについてＬＵ分解を行ったあと，上記のような行列積を行う。このとき，行列Ｃをｎ個に分割する。それを順にＣ１，Ｃ２，Ｃ３，…，Ｃｎとする。ｎは，＃ｐｅ／ｎ＞１で，ＬＯＧ２（＃ｐｅ／ｎ）＜ｎとなるように決める。

【００３９】図７に示すように，８個のＰＥ１～ＰＥ８で４分割した行列Ｃ１～Ｃ４について計算を行う場合を例に説明する。ＰＥ１がブロックＲ１のデータを，ＰＥ２がブロックＲ２のデータを，…，ＰＥ８がブロックＲ８のデータを保持していたとする。各ＰＥの行列Ｒｉと行列Ｃｊの積の行列積で部分的に更新を行う。図７に示すハッチングの部分は，第１回目の更新を行う部分を表す。

【００４０】各ＰＥの行列Ｒｉと行列積を行う行列Ｃｊは，ＰＥ１からＰＥ８まで順に表すと，次のとおりである。

１回目の計算：Ｃ１，Ｃ２，Ｃ３，Ｃ４，Ｃ１，Ｃ２，Ｃ３，Ｃ４

２回目の計算：Ｃ４，Ｃ１，Ｃ２，Ｃ３，Ｃ４，Ｃ１，Ｃ２，Ｃ３

３回目の計算：Ｃ３，Ｃ４，Ｃ１，Ｃ２，Ｃ３，Ｃ４，Ｃ１，Ｃ２

４回目の計算：Ｃ２，Ｃ３，Ｃ４，Ｃ１，Ｃ２，Ｃ３，Ｃ４，Ｃ１

このような計算を行うためには，Ｃｋの転送を行う必要がある。このため，次のように転送を行う。

【００４１】１回目の転送でブロックｋがあるプロセッサはｐ＝ｍｏｄ（ｋ－１，＃ｐｅ）＋１である。このプロセッサｐから，行列Ｃをｎ分割したＣｉをｍｏｄ（ｐ－２＋ｉ，＃ｐｅ）＋１へ転送する。２回目からは，１回目の転送で転送されたｎ個のプロセッサのデータを並列に残りのプロセッサに転送する。順次ｔ回目の転送では２＊＊（ｔ－１）＊ｎ（なお，＊＊は巾乗を表す）のデータを使って並列に転送する。このようにして各プロセッサにＣｉを転送する。

【００４２】図８は，ＰＥ１に行列Ｃがあった場合の転送例を示す。図８に示すように，１回目の転送でＣ１はＰＥ１へ，Ｃ２はＰＥ２へ転送される。２回目の転送では，Ｃ１はＰＥ１からＰＥ３へ，同時にＣ２はＰＥ２からＰＥ４へ並列に転送される。次の転送では，Ｃ１はＰＥ１からＰＥ５へ，ＰＥ３からＰＥ７へ転送され，同時にＣ２はＰＥ２からＰＥ６へ，ＰＥ４からＰＥ８へ並列に転送される。

【００４３】ここで，２回目の計算で必要なデータは，１回目の計算を行っている間に，別の領域に転送しておくことにより，転送と計算とを同時に行うことができる。図９および図１０は，行列Ｃがブロックｋつまりｐ＝ｍｏｄ（ｋ－１，＃ｐｅ）＋１にあったときでｎが偶数の場合の処理フローを示す。図９および図１０に示す処理フローチャートでは，１回目の計算，つまり奇数回目の計算では第１のワーク領域（Ｗ１），偶数回目の計算では第２のワーク領域（Ｗ２）を用いて，計算を行う。

【００４４】まず，ステップＳ２１において，Ｃｉ（ｉ＝１，…，ｎ）をプロセッサｍｏｄ（ｐ＋ｉ－２，＃ｐｅ）＋１のワーク領域Ｗ１に転送し，＄ｅ＝０とする。ステップＳ２２において，＄ｎ＝Ｎ＊２＊＊＄ｅ，＄ｔ＝ｍｉｎ（＄ｎ，＃ｐｅ－＄ｎ）とする。ｍｉｎは，最小値を得る関数である。

【００４５】ステップＳ２３において，＄ｔ個のプロセッサからＣｉをワーク領域Ｗ１に転送する。また，ｓ＝

ｍｏｄ（ｐ＋ｊ－２，＃ｐｅ）＋１，ｄ＝ｍｏｄ（ｐ＋＄ｔ＋ｊ－２，＃ｐｅ）＋１とし，プロセッサｓからプロセッサｄにｊ＝１，…，＄ｔの＄ｔ個を並列に転送する。

【００４６】ステップＳ２４において，＄ｅ＝＄ｅ＋１とする。ステップＳ２５において，＄ｎ＞＄ｔかどうかを判定し，＄ｎが＄ｔより大きければステップＳ２２へ戻り，＄ｎが＄ｔより小さければステップＳ２６の処理を行う（図１０）。

【００４７】ステップＳ２６において，ｃｔ＝１とする。ステップＳ２７において，ｃｔ＝＝１であるかどうかを判定する。ｃｔ＝＝１であればステップＳ２９へ進み，ｃｔ＝＝１でなければステップＳ２８の処理を行う。

【００４８】ステップＳ２８において，後述するステップＳ３３の処理の終了を待って，プロセッサｐについてのみ，Ｃｉ（ｉ＝ｃｔ）をデータとしてワーク領域Ｗ１へ転送する。

【００４９】ステップＳ２９において，プロセッサｉからプロセッサｍｏｄ（Ｉ，＃ｐｅ）＋１にデータを転送する（Ｗ１からＷ２への転送）。ステップＳ３０において，開始後，各プロセッサにあるＣｉのデータで対応する部分の行列の更新を並列に行う（Ｗ１を使って計算）。

【００５０】ステップＳ３１において，ｃｔ＞１であるかどうかを判定する。ｃｔ＞１であればステップＳ３３の処理へ進み，ｃｔ＞１でなければステップＳ３２の処理を行う。

【００５１】ステップＳ３２において，ステップＳ２９の処理（Ｗ１からＷ２への転送）の終了を待つ。ステップＳ３３において，ｃｔ＝ｃｔ＋１とする。プロセッサｐについてのみ，Ｃｉ（ｉ＝ｃｔ）をデータとしてワーク領域Ｗ２へ転送する。

【００５２】ステップＳ３４において，ｃｔ＝＝ｎであるかどうかを判定する。ｃｔ＝＝ｎであればステップＳ３６の処理へ進み，ｃｔ＝＝ｎでなければステップＳ３５の処理を行う。

【００５３】ステップＳ３５において，プロセッサｉからプロセッサｍｏｄ（Ｉ，＃ｐｅ）＋１にデータを転送する（Ｗ２からＷ１への転送）。ステップＳ３６において，ワーク領域Ｗ２のデータを使って対応する行列の更新を各プロセッサで並列に行う。

【００５４】ステップＳ３７において，ｃｔ＝ｃｔ＋１とする。ステップＳ３８において，ｃｔ＞ｎであるかどうかを判定する。ｃｔ＞ｎであれば処理を終了し，ｃｔ＞ｎでなければステップＳ２７の処理へ戻る。

【００５５】行列Ｃの分割方法について行方向の分割を説明したが，列方向に分割しても同様に処理を行うことができる。ただし，並列化とベクトル化のバランスを考えると，適当なベクトル長でブロック幅を持たせることのできる行方向での分割のほうが好ましい。

【００５６】この効果は，行列Ｃ全体を２分木の方法で，各ＰＥに転送した場合，～ＬＯＧ２（＃ｐｅ）のオーダの転送時間がかかるのに対して，～１－（ＬＯＧ２（＃ｐｅ／ｎ））／ｎのオーダとなり＃ｐｅ数が大きいときは，非常に高速である。

【００５７】

〔３〕前進／後進代入を並列に行う上での方式

ＬＵ分解を行った後での前進／後進代入にも高速化のためには並列性が必要である。この並列性を引き出すために，次のように行う。

【００５８】第１に，ＬＵ分解を行うときに図５に示すようにブロックを各ＰＥに対してサイクリックに割り当てているので，これを元の割り付け方法に戻す。次に，元の行列を列方向に分割していたのを，行方向に分割したデータ配置に変更し，この配置をもとに前進／後進代入を並列に行う。

【００５９】すなわち，ＬＵ分解を行うときには，図１１（Ａ）の行列Ａのように，列ベクトルを各ＰＥに分散して配置している。これを前進／後進代入を並列に実行できるように，図１１（Ｂ）の行列Ｂのような配置に変更し，行ベクトルを各ＰＥに配置する。これを並列に実行して並べ換える。

【００６０】この変換を並列に行うために，行列を各ＰＥに分散配置される境界で図１２に示すように分割して，

Ａ＝（ａｉｊ）　〔ｉ＝１，…，＃ｐｅ，ｊ＝１，…，＃ｐｅ〕

とする。なお，＃ｐｅはプロセッサ数であり，図１２はプロセッサ数＃ｐｅが５である場合を示している。

【００６１】行ベクトルを各ＰＥに割り付けた行列Ａと，同じ大きさの行列を列ベクトルで割り付けた行列Ｂとの間で，並べ換えのためのデータ転送を，図１２に示すハッチング部分のような，対角方向のブロック要素について行う。

【００６２】前述のように，図２に示す本実施例のメモリ分散型並列計算機では，各プロセッサに対して同時に１つの読み込みと１つの書き込みが可能である。図１３に，図１１に示す行列Ａから行列Ｂへの変換の処理フローチャートを示す。

【００６３】図１３のステップＳ４１において，各プロセッサ（１～＃ｐｅ）で，ｋ＝プロセッサ番号，ｊ＝ｋ，＃ｃｔ＝１とする。ステップＳ４２において，並列に各プロセッサでＢｊｋ＝Ａｊｋのデータの配置替えを行う。

【００６４】ステップＳ４３において，ｋ＝ｍｏｄ（ｋ，＃ｐｅ）＋１，＃ｃｔ＝＃ｃｔ＋１とする。ステップＳ４４において，＃ｃｔ＞＃ｐｅであるかどうかを判定する。＃ｃｔ＞＃ｐｅであれば処理を終了し，＃ｃｔ＞＃ｐｅでなければステップＳ４２の処理へ戻る。

【００６５】行列Ｂは，図１４に示すように，行方向に分割配置されている。ここで，行列ＢはＬＵ分解できたとする。ＬＵｘ＝ｄを解くとき，Ｌｙ＝ｄを解き，Ｕｘ＝ｙを順に解く。これを並列に行うために，各ＰＥにｄ，ｘ，ｙを重複して持つ。Ｕｘ＝ｙについても同様に行うことができるので，Ｌｙ＝ｄについて説明する。

【００６６】まず，ＰＥ１でL_{11}×ｙ１＝ｄ１を解く。ＰＥ１のｙ１を各プロセッサ上の変数ｙのｙ１の部分へ２の巾乗パターンで転送する。（＃ｐｅ－１）個のＰＥで並列にｄｉ＝ｄｉ－L_{i1}×ｙ１を行う（ｉ＝２，…，＃ｐｅ）。

【００６７】同様に，ＰＥ２でL_{22}×ｙ１＝ｄ１を解く。ＰＥ２のｙ２を各プロセッサ上の変数ｙのｙ２の部分へ２の巾乗パターンで転送する。（＃ｐｅ－２）個のＰＥで並列にｄｉ＝ｄｉ－L_{i2}×ｙ２を行う（ｉ＝３，…，＃ｐｅ）。

【００６８】同様に，ＰＥｋでL_{kk}×ｙｋを解く。ＰＥｋのｙｋを各プロセッサ上の変数ｙのｙｋの部分へ２の巾乗パターンで転送する。（＃ｐｅ－ｋ）個のＰＥで並列にｄｉ＝ｄｉ－L_{k2}×ｙｋを行う（ｉ＝ｋ，…，＃ｐｅ）。

【００６９】最後に，L_{55}×ｙ５＝ｄ５を解いて，ｙ５を各プロセッサ上の変数ｙのｙ５の部分へ２の巾乗パターンで転送する。結果として，各プロセッサに解ｙが求まる。

【００７０】次に，２００×２００の行列を５プロセッサで解く場合を例にして，本発明の適用例を詳しく説明する。ブロック幅を１０と仮定する。すなわち，この例では全部で２０ブロック×２０ブロックの行列となっている。

【００７１】この行列を各ＰＥに配置すると，プロセッサ数が５であるので，各ＰＥが担当する部分はそれぞれ２０ブロック÷５の４ブロックの列となる。これらのブロックを，図１５（Ａ）に示すように，ＰＥ１から順にブロック１，ブロック２，ブロック３，…，ブロック２０とする。

【００７２】並列に計算を実行する部分を各ＰＥに均等に割り付けるために，図１５（Ａ）に示すデータ配置を，図１５（Ｂ）に示すように並べ換える。ここでは，各ブロックをブロック・サイクリック（ｂｌｏｃｋ　ｃｙｃｌｉｃ）に並べ換えている。この並べ換えでは，例えばブロック２，６，１０，１４，１８の転送を同時に行い，読み込みと書き込みとが各ＰＥでそれぞれ行われるようにして，並列転送を実現する。

【００７３】並べ換えの結果，図１５（Ｂ）に示すように，ＰＥ１の行列はブロック１，６，１１，１６の並びとなり，以下ＰＥ２はブロック２，７，１２，１７，ＰＥ３はブロック３，８，１３，１８，ＰＥ４はブロック４，９，１４，１９，ＰＥ５はブロック５，１０，１５，２０となる。

【００７４】最初に，ブロック１をＬＵ分解する。ブロック１（図１６のｂ１）は，ＰＥ１だけで計算する。図１６のハッチング部分が計算完了となる。図１６はブロックｂ１と各ＰＥの転送先となるワーク領域の関係を示す。各ＰＥは第１および第２のワーク領域を持つ。ＰＥ１の第１ワーク領域をＷ１１，第２ワーク領域をＷ１２，ＰＥ２の第１ワーク領域をＷ２１，第２ワーク領域をＷ２２，ＰＥ３の第１ワーク領域をＷ３１，第２ワーク領域をＷ３２，ＰＥ４の第１ワーク領域をＷ４１，第２ワーク領域をＷ４２，ＰＥ５の第１ワーク領域をＷ５１，第２ワーク領域をＷ５２とする。

【００７５】図１６にハッチングで示したブロックｂ１をＣ１～Ｃ３に３等分した場合を考える。この部分の計算が完了したならば，Ｃ１をＰＥ１のＷ１１へ，Ｃ２をＰＥ２のＷ２１へ，Ｃ３をＰＥ３のＷ３１へそれぞれ転送する。次に，その結果を使って，Ｗ１１（Ｃ１）のデータをＰＥ４のＷ４１へ，Ｗ２１（Ｃ２）のデータをＰＥ５のＷ５１へ並列転送する。

【００７６】行列積の計算では，初めに各ＰＥの第１のワーク領域（Ｗ１１，Ｗ２１，Ｗ３１，Ｗ４１，Ｗ５１）に格納されたＣｉを使って計算を行う。図１７に示すハッチング部分が最初に計算する部分である。ＰＥ１においてＣ１×Ｒ１の行列積が，ＰＥ２においてＣ２×Ｒ２の行列積が，ＰＥ３においてＣ３×Ｒ３の行列積が，ＰＥ４においてＣ１×Ｒ４の行列積が，ＰＥ５においてＣ２×Ｒ５の行列積がそれぞれ計算されることになる。

【００７７】これらの計算と同時にオーバーラップしてＷ１１からＷ２２へ，Ｗ２１からＷ３２へ，Ｗ３１からＷ４２へ，Ｗ４１からＷ５２へ，並列にデータ転送を行うとともに，ＰＥ１が保持するブロックｂ１のＣ３をＷ１２に転送する。

【００７８】次に，各ＰＥの第２のワーク領域（Ｗ１２，Ｗ２２，Ｗ３２，Ｗ４２，Ｗ５２）に格納されたＣｉを使って計算を行う。図１８に示すハッチッグ部分が次に計算する部分である。ＰＥ１においてＣ３×Ｒ１の行列積が，ＰＥ２においてＣ１×Ｒ２の行列積が，ＰＥ３においてＣ２×Ｒ３の行列積が，ＰＥ４においてＣ３×Ｒ４の行列積が，ＰＥ５においてＣ１×Ｒ５の行列積がそれぞれ計算されることになる。

【００７９】これらの計算と同時にオーバーラップしてＷ１２からＷ２１へ，Ｗ２２からＷ３１へ，Ｗ３２からＷ４１へ，Ｗ４２からＷ５１へ，並列にデータ転送を行うとともに，ＰＥ１が保持するブロックｂ１のＣ２をＷ１１に転送する。

【００８０】３回目の計算では，各ＰＥの第１のワーク領域（Ｗ１１，Ｗ２１，Ｗ３１，Ｗ４１，Ｗ５１）に格納されたＣｉを使って計算を行う。図１９に示すハッチッグ部分が計算する部分である。ＰＥ１においてＣ２×Ｒ１の行列積が，ＰＥ２においてＣ３×Ｒ２の行列積

が，ＰＥ３においてＣ１×Ｒ３の行列積が，ＰＥ４においてＣ２×Ｒ４の行列積が，ＰＥ５においてＣ３×Ｒ５の行列積がそれぞれ計算されることになる。

【００８１】図１５（Ｂ）に示すブロック２（図２０のｂ２）に関するＬＵ分解と対応する行列積の計算は，図２０（Ａ）に示すＰＥ２のハッチング部分をＬＵ分解し，計算に必要な部分をブロックｂ１と同じようにＣ１～Ｃ３に３等分して転送することにより行う。

【００８２】図２０（Ｂ）はブロックｂ２と各ＰＥの転送先となるワーク領域の関係を示す。Ｃ１をＰＥ２のＷ２１へ，Ｃ２をＰＥ３のＷ３１へ，Ｃ３をＰＥ４のＷ４１へそれぞれ転送する。次にその結果を使って，Ｗ２１（Ｃ１）のデータをＰＥ５のＷ５１へ，Ｗ３１（Ｃ２）のデータをＰＥ１のＷ１１へ並列転送する。

【００８３】次に，各ＰＥの第１のワーク領域（Ｗ１１，Ｗ２１，Ｗ３１，Ｗ４１，Ｗ５１）に格納されたＣｉを使って計算を行う。図２１に示すハッチング部分が今回計算する部分である。ＰＥ１においてＣ２×Ｒ１の行列積が，ＰＥ２においてＣ１×Ｒ２の行列積が，ＰＥ３においてＣ２×Ｒ３の行列積が，ＰＥ４においてＣ３×Ｒ４の行列積が，ＰＥ５においてＣ１×Ｒ５の行列積がそれぞれ計算されることになる。

【００８４】これらの計算と同時にＷ２１からＷ３２へ，Ｗ３１からＷ４２へ，Ｗ４１からＷ５２へ，Ｗ５１からＷ１２へ並列にデータ転送を行うとともに，ＰＥ２が保持するブロックｂ２のＣ３をＷ２２に転送する。

【００８５】以下，図１８および図１９を用いて説明したブロックｂ１の場合と同様に計算と転送を行い，ブロックｂ２についての計算が終了すると，図１５（Ｂ）に示すブロック３（図２２のｂ３）に関するＬＵ分解と対応する行列積の計算を行う。図２２（Ａ）に示すブロックｂ３をＬＵ分解したあと，図２２（Ｂ）に示すようにブロックｂ３をＣ１～Ｃ３に３等分し，各ＰＥのワーク領域への転送を行う。そして，ブロックｂ３に関しても，図１７ないし図１９で説明したのと同様に計算と転送を行い，以下，ブロック４，５，…，２０まで同様に処理してＬＵ分解を完了する。

【００８６】全てのブロックについてＬＵ分解を行った後，最終処理として，サイクリックに並べ換えたブロックを元の並びに戻す。図２３は，図１５に示すようにブロック・サイクリックに並べ換えた行列を元の並びに戻す例を示している。例えばブロック２，６，１０，１４，１８に着目すると，図２３（Ａ）から（Ｂ）への転送例から明らかなように，これらのブロックは同時に転送が可能である。

【００８７】次に列方向に分割して配置していたものを行方向に分割する並びに並べ換えて，並列に前進／後退代入を行う。図２４は，ブロック化された行列の対角ブロック方向の要素ブロックに着目して，並列転送で並べ換える例を示す。ＰＥ１のＡ$_{51}$をＰＥ５の１番目の領域へ，ＰＥ２のＡ$_{12}$をＰＥ１の２番目の領域へ，……，というように並列転送を行って，ブロックを並べ換える。ブロックの並べ換えにより，配置が変わったところで，前進／後退代入を行う。

【００８８】図２５は，行方向に分割された行列と各ＰＥの関係を示す。解ベクトルを求めるためのｂ$_1$，ｂ$_2$，ｂ$_3$，ｂ$_4$，ｂ$_5$ は各ＰＥで重複して保持する。計算が終了した時点で各ＰＥが解ベクトルを持つことになる。計算手順は以下のとおりである。

【００８９】（１）Ｌ$_{11}$ｘ$_1$ ＝ｂ$_1$ をＰＥ１で解く。

（２）ｘ$_1$ をｂ$_1$ の領域に求めて，各ＰＥへ転送する。ｂ$_1$ ＝ｘ$_1$

（３）ｉ＞１なるＰＥｉでｂ$_i$ ＝ｂ$_i$ －Ｌ$_{i1}$×ｘ$_1$ を計算する。

【００９０】（４）次にＰＥ２で，Ｌ$_{22}$×$_2$ ＝ｂ$_2$ を解いてｘ$_2$ を求める。

（５）ｘ$_2$ を各ＰＥへ転送する。ｂ$_2$ ＝ｘ$_2$

（６）ｉ＞２なるＰＥｉでｂ$_i$ ＝ｂ$_i$ －Ｌ$_{i2}$×ｘ$_2$ を計算する。

【００９１】以下，同様に繰り返して前進代入を終了する。前進代入のあと，後進代入も同様に行う。

【００９２】

【発明の効果】以上説明したように，本発明によれば，次のような効果がある。

① データの並べ換えを動的に行うことにより，並列に実行する部分を各プロセッサに均等に割り付けることができるようになり，並列に実行する上での効率が向上する。実際に，ブロック化した外積型のガウスの消去法をもとにしたＬＵ分解の行列積の部分に対しては，行列を列ベクトル方向に均等に分割して計算した場合の実効性能は，ハードウェア性能の６割５分程度である。これは計算過程が進むと行列積で更新する部分が小さくなり，配置されているプロセッサ数が急激に減少するので，並列効率が悪くなるためである。

【００９３】これに対して，データをサイクリックに分割すると，つまり列ベクトルを束ねたブロックに番号を振り，その番号をｉとしたとき，ｉ番目のブロックが，ｍｏｄ（Ｉ－１，＃ｐｅ）＋１番目のプロセッサ（＃ｐｅはプロセッサ数）に割り付けられるように配置した場合には，実行性能は，ハードウェア性能の９割～９割５分程度が達成される。

【００９４】② 行列積の部分の計算方法に関しては，行列積部分の計算に必要なデータを分割して，転送・計算する。このとき，本発明によれば転送の大部分を計算と同時に行うことができ，計算と同時に行うことのできない最初の転送時間だけが見かけ上の転送時間となる。この転送も並列に行う工夫により，例えば２分木転送に比べて，大幅に転送時間を短縮することができる。この結果，単純に行列積で必要なデータを各プロセッサに２分木のパターンで転送する場合がＬＯＧ２（＃ｐｅ）に

比例するのに比べ，転送時間は，転送するデータの分割数を＃ｄｉｖとすると，１＋（ＬＯＧ２（＃ｐｅ／＃ｄｉｖ）／＃ｄｉｖ）のオーダになり，第２項は０．５以下にすることができる。したがって，２台以上あるシステムで特にプロセッサ数が大きくなった場合に非常に効率がよい。

【００９５】③　また，最後に前進／後進代入をこのままの配置で解くと，列ベクトルは１つのプロセッサ上にあるため，前進／後進代入部分の並列性が利用できず，並列化できない。このため，サイクリックなデータの配置を，一度，列ベクトル方向に均等分割する配置に戻し，その後，行ベクトル方向に均等分割する配置に変える。このことにより，前進／後進代入部分を並列に実行することができるようになり，処理時間の大幅な短縮が可能になる。

【図面の簡単な説明】

【図１】本発明の原理説明図である。

【図２】本発明の実施例に係るメモリ分散型並列計算機の例を示す図である。

【図３】図２に示すプロセッサの構成を示す図である。

【図４】本発明の実施例における並べ換えの処理フローを示す図である。

【図５】本発明の実施例におけるブロックの転送例を示す図である。

【図６】本発明の実施例におけるＬＵ分解の対象となる行列の例を示す図である。

【図７】本発明の実施例における行列積の計算を説明する図である。

【図８】ＰＥ１に行列Ｃがあった場合の転送例を示す図である。

【図９】データの転送と行列積の計算の処理フローを示す図である。

【図１０】データの転送と行列積の計算の処理フローを示す図である。

【図１１】前進／後進代入処理時の並べ換えの例を示す図である。

【図１２】前進／後進代入処理時の並べ換えを説明する図である。

【図１３】前進／後進代入処理時の並べ換えの処理フローを示す図である。

【図１４】前進／後進代入処理を説明する図である。

【図１５】本発明の適用例の説明図である。

【図１６】本発明の適用例の説明図である。

【図１７】本発明の適用例の説明図である。

【図１８】本発明の適用例の説明図である。

【図１９】本発明の適用例の説明図である。

【図２０】本発明の適用例の説明図である。

【図２１】本発明の適用例の説明図である。

【図２２】本発明の適用例の説明図である。

【図２３】本発明の適用例の説明図である。

【図２４】本発明の適用例の説明図である。

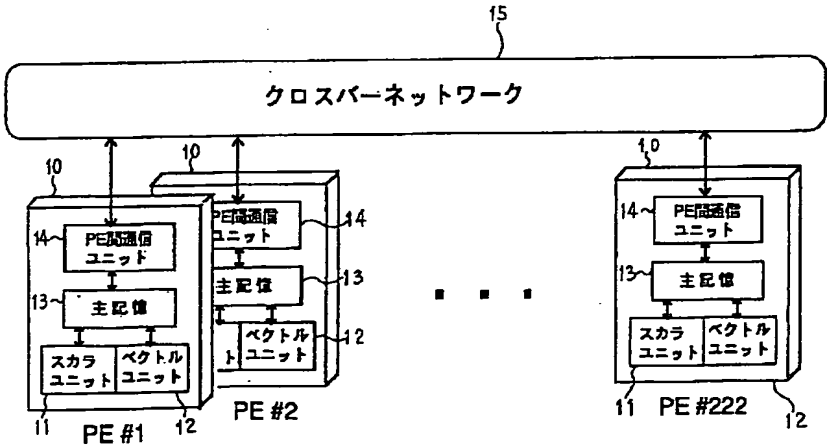
【図２５】本発明の適用例の説明図である。

【図２６】ブロック化した外積型のＬＵ分解法の説明図である。

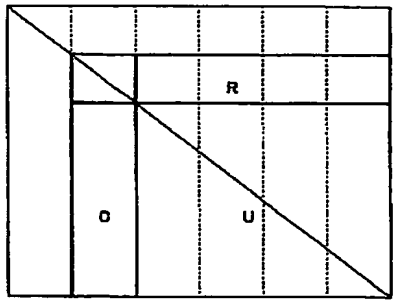
【符号の説明】

１　メモリ分散型並列計算機
２　データ再配置処理手段
３　ＬＵ分解処理手段
４　前進／後進代入処理手段

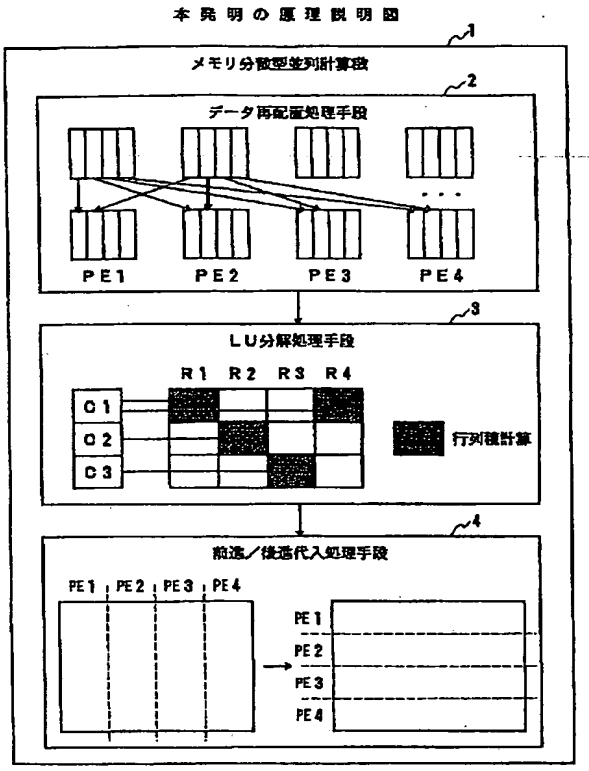
【図２】



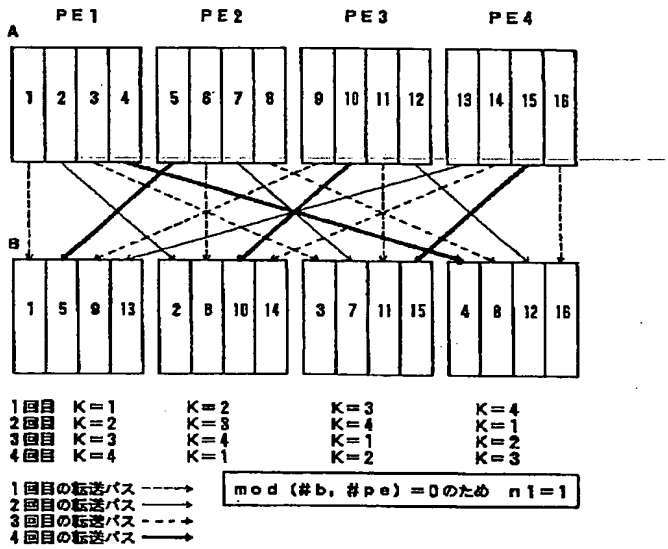
【図６】



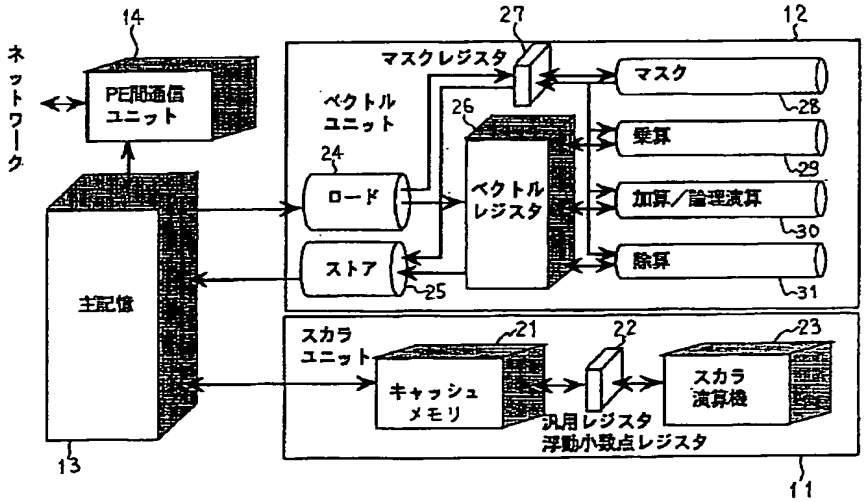
【図１】


本発明の原理説明図
メモリ分散型並列計算機
データ再配置処理手段
PE1
PE2
PE3
PE4
LU分解処理手段
R1 R2 R3 R4
C1
C2
C3
行列積計算
前進／後退代入処理手段
PE1 PE2 PE3 PE4

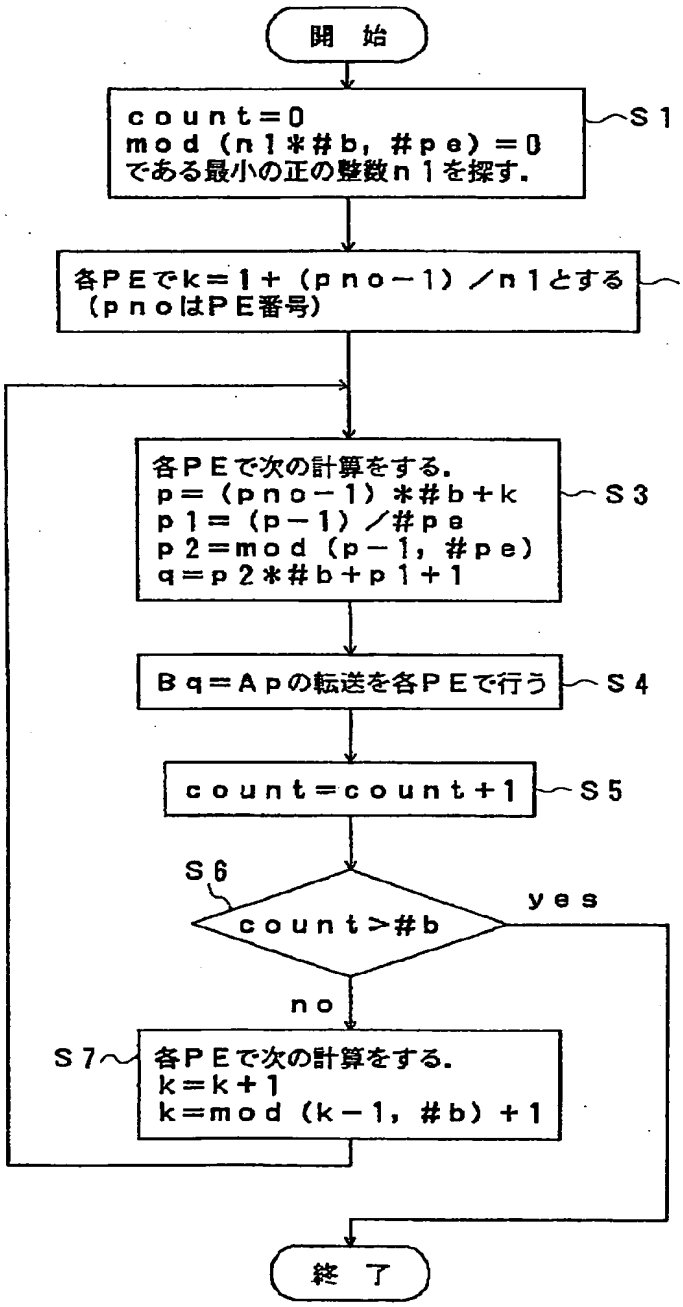
【図５】


PE1
PE2
PE3
PE4
A
B
1回目 K=1 K=2 K=3 K=4
2回目 K=2 K=3 K=4 K=1
3回目 K=3 K=4 K=1 K=2
4回目 K=4 K=1 K=2 K=3
1回目の転送パス
2回目の転送パス
3回目の転送パス
4回目の転送パス
mod (#b, #pe) =0のため n1=1

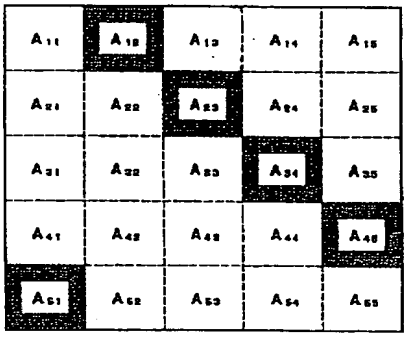
【図３】


ネットワーク
PE間通信ユニット
主記憶
ベクトルユニット
マスクレジスタ
ロード
ストア
ベクトルレジスタ
マスク
乗算
加算／論理演算
除算
スカラユニット
キャッシュメモリ
スカラ演算機
汎用レジスタ
浮動小数点レジスタ

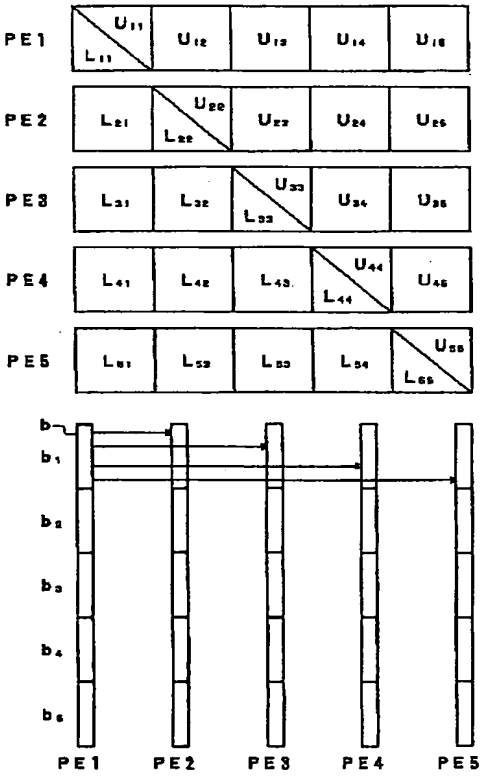
【図４】


開　始
ｃｏｕｎｔ＝０
ｍｏｄ（ｎ１＊＃ｂ，＃ｐｅ）＝０
である最小の正の整数ｎ１を探す．
Ｓ１
各ＰＥでｋ＝１＋（ｐｎｏ－１）／ｎ１とする
（ｐｎｏはＰＥ番号）
Ｓ２
各ＰＥで次の計算をする．
ｐ＝（ｐｎｏ－１）＊＃ｂ＋ｋ
ｐ１＝（ｐ－１）／＃ｐｅ
ｐ２＝ｍｏｄ（ｐ－１，＃ｐｅ）
ｑ＝ｐ２＊＃ｂ＋ｐ１＋１
Ｓ３
Ｂｑ＝Ａｐの転送を各ＰＥで行う
Ｓ４
ｃｏｕｎｔ＝ｃｏｕｎｔ＋１
Ｓ５
Ｓ６
ｃｏｕｎｔ＞＃ｂ
ｙｅｓ
ｎｏ
Ｓ７
各ＰＥで次の計算をする．
ｋ＝ｋ＋１
ｋ＝ｍｏｄ（ｋ－１，＃ｂ）＋１
終　了

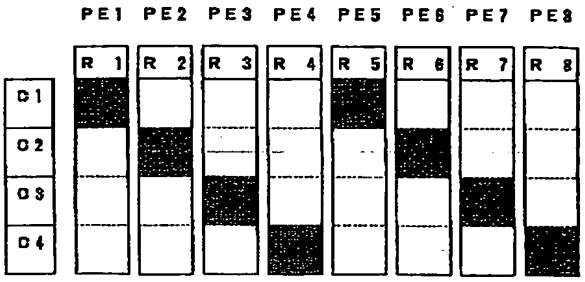
【図１２】



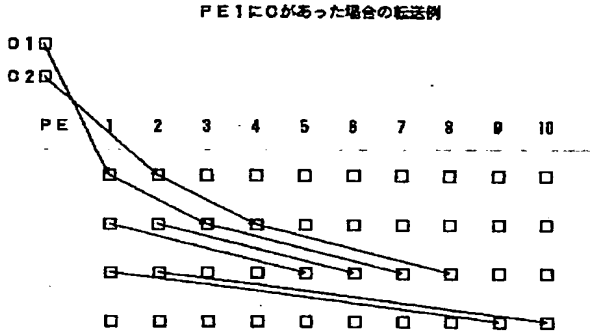
【図２５】


PE1
PE2
PE3
PE4
PE5
b
b1
b2
b3
b4
b5
PE1
PE2
PE3
PE4
PE5

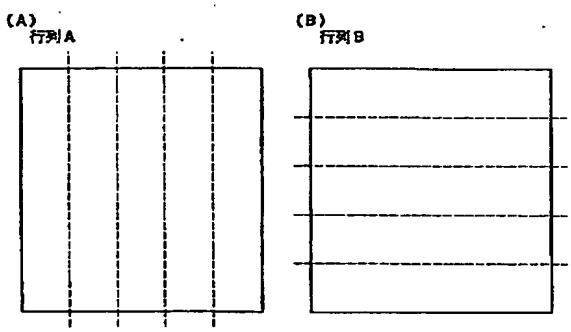
【図７】


PE1 PE2 PE3 PE4 PE5 PE6 PE7 PE8
R 1
R 2
R 3
R 4
R 5
R 6
R 7
R 8
O1
O2
O3
O4

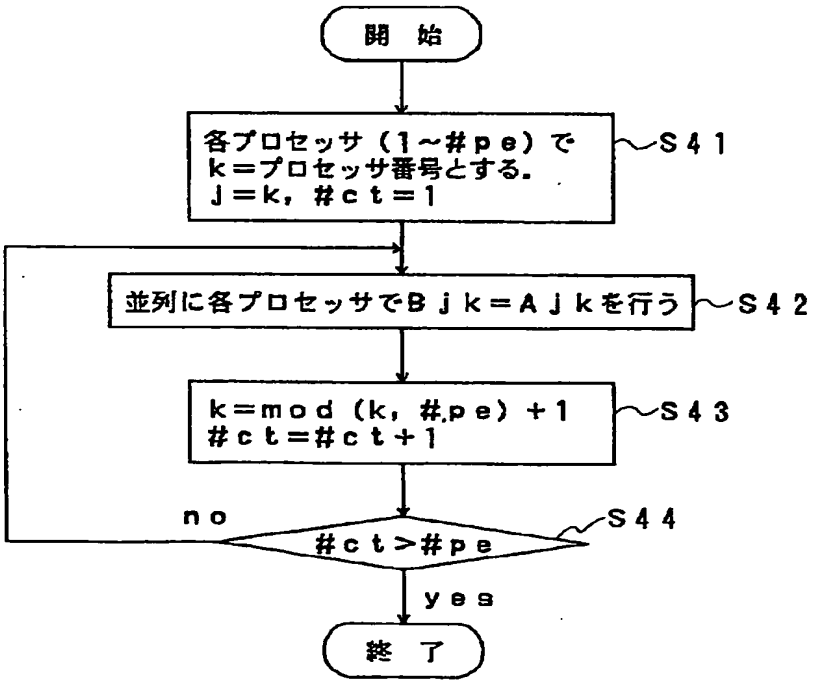
【図８】


PE1にOがあった場合の転送例
O1
O2
PE 1 2 3 4 5 6 7 8 9 10

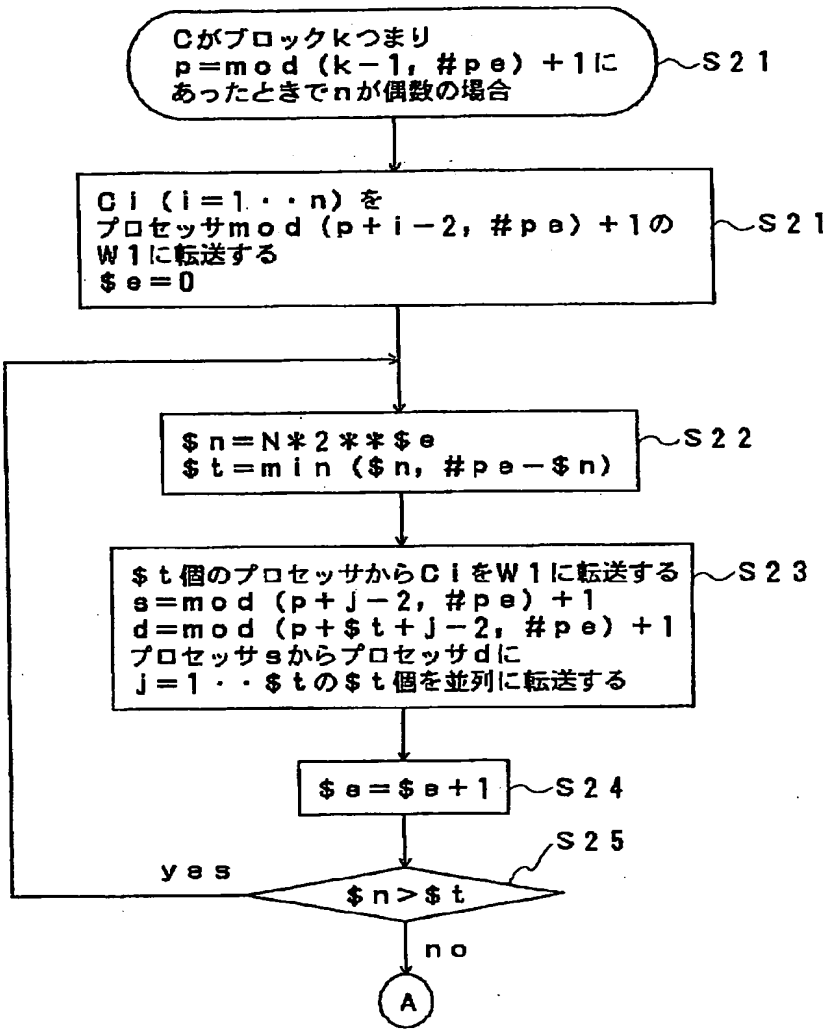
【図１１】


(A)
行列A
(B)
行列B

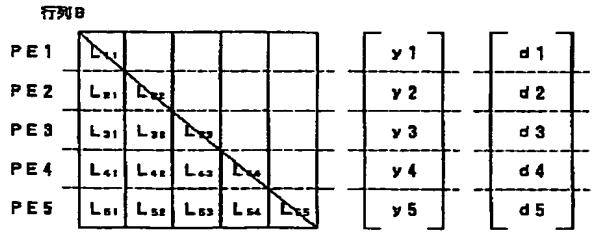
【図１３】


開　始
各プロセッサ（1～#ｐｅ）で
ｋ＝プロセッサ番号とする。
Ｊ＝ｋ，＃ｃｔ＝1
S41
並列に各プロセッサでＢｊｋ＝Ａｊｋを行う
S42
ｋ＝ｍｏｄ（ｋ，＃ｐｅ）＋1
＃ｃｔ＝＃ｃｔ＋1
S43
no
＃ｃｔ＞＃ｐｅ
S44
ｙｅｓ
終　了

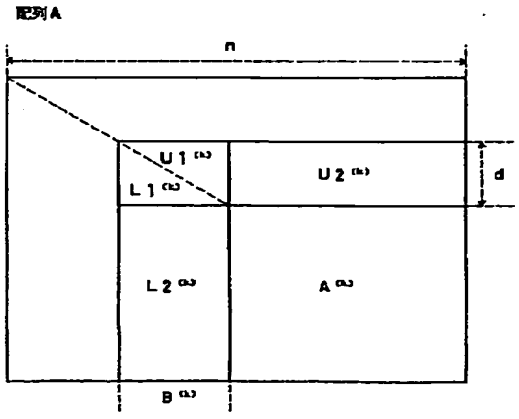
【図９】


Cがブロックkつまり
p＝mod（k－1，#pe）＋1に
あったときでnが偶数の場合
S21
Ci（i＝1・・n）を
プロセッサmod（p＋i－2，#pe）＋1の
W1に転送する
$e＝0
S21
$n＝N＊2＊＊$e
$t＝min（$n，#pe－$n）
S22
$t個のプロセッサからCiをW1に転送する
s＝mod（p＋j－2，#pe）＋1
d＝mod（p＋$t＋j－2，#pe）＋1
プロセッサsからプロセッサdに
j＝1・・$tの$t個を並列に転送する
S23
$e＝$e＋1
S24
S25
yes
$n>$t
no
A

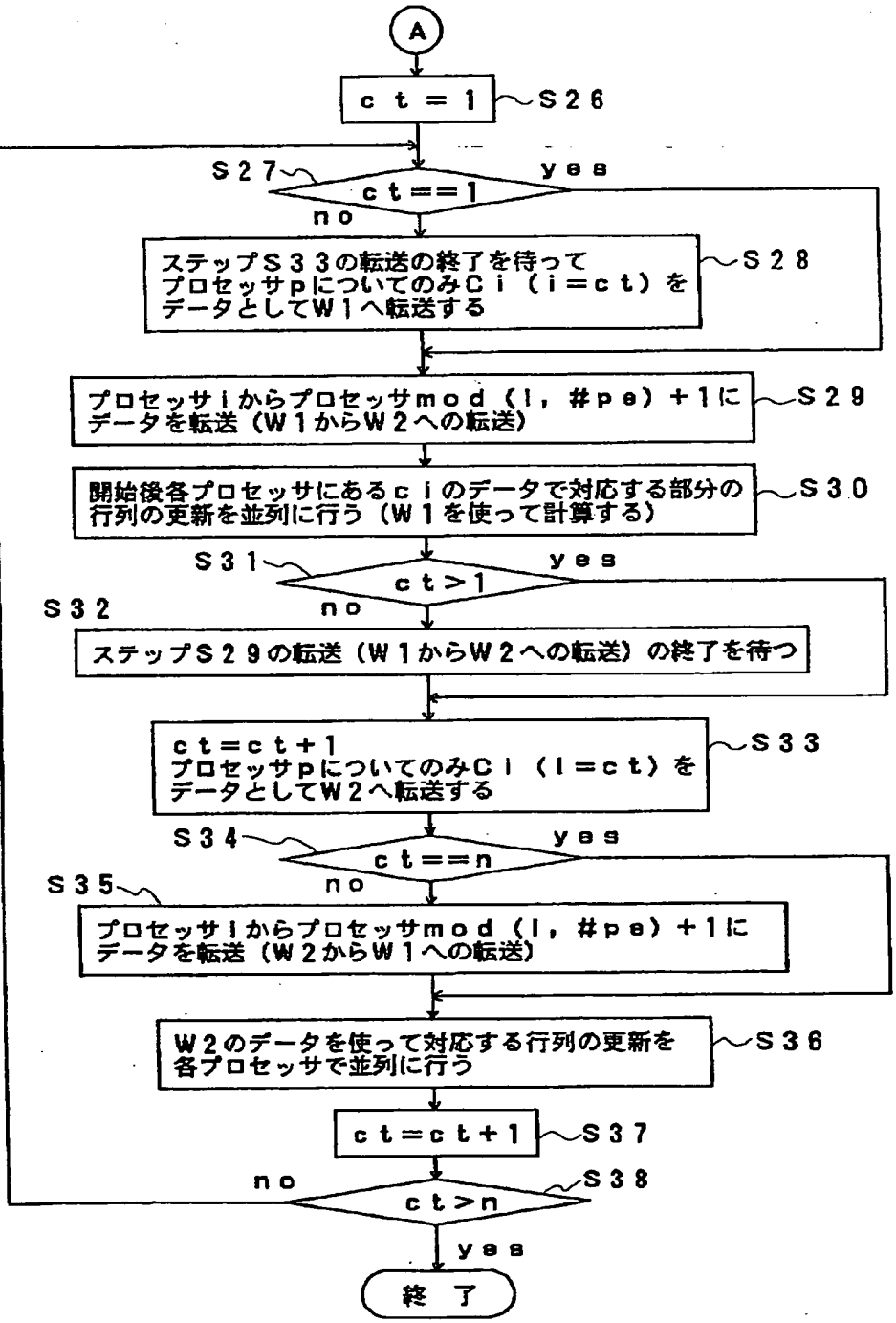
【図１４】


行列B
PE1
PE2
PE3
PE4
PE5
y1
y2
y3
y4
y5
d1
d2
d3
d4
d5

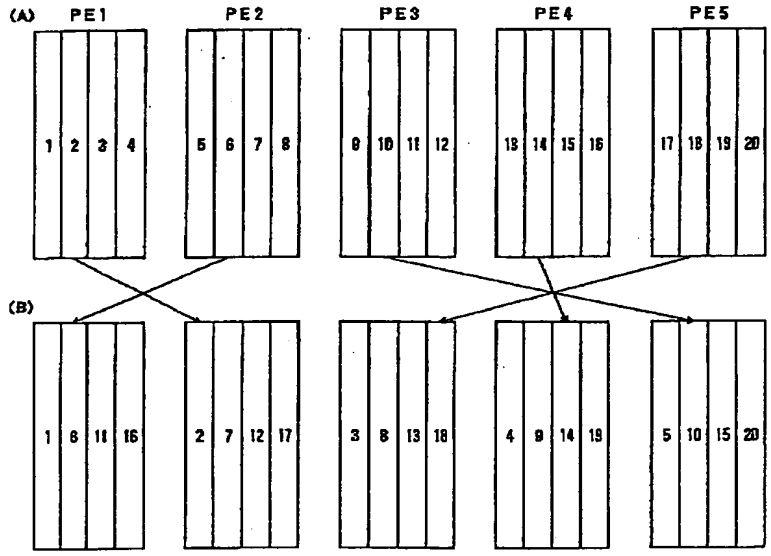
【図２６】


配列A
n
U1(3)
L1(3)
U2(3)
d
L2(3)
A(3)
B(3)

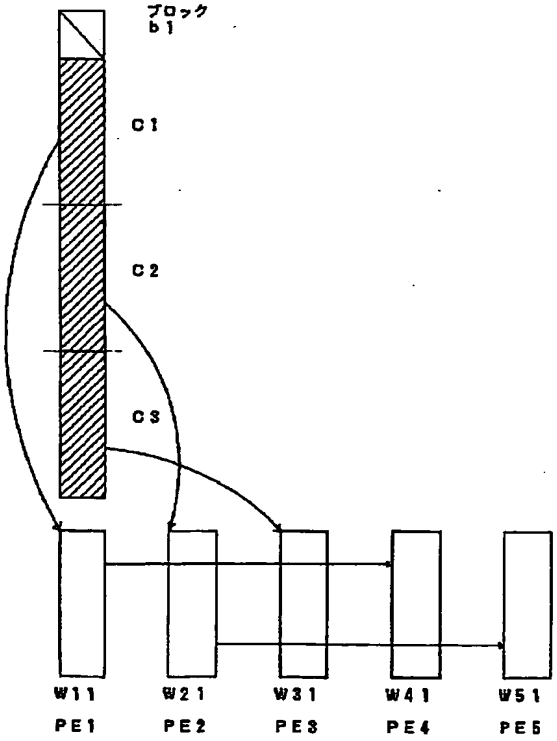
【図１０】


A
ｃｔ＝１
S26
S27
ｃｔ==１
yes
no
ステップＳ３３の転送の終了を待って
プロセッサｐについてのみＣｉ（ｉ＝ｃｔ）を
データとしてＷ１へ転送する
S28
プロセッサｉからプロセッサｍｏｄ（ｉ，＃ｐｅ）＋１に
データを転送（Ｗ１からＷ２への転送）
S29
開始後各プロセッサにあるｃｉのデータで対応する部分の
行列の更新を並列に行う（Ｗ１を使って計算する）
S30
S31
ｃｔ＞１
yes
no
S32
ステップＳ２９の転送（Ｗ１からＷ２への転送）の終了を待つ
ｃｔ＝ｃｔ＋１
プロセッサｐについてのみＣｉ（ｉ＝ｃｔ）を
データとしてＷ２へ転送する
S33
S34
ｃｔ==ｎ
yes
no
S35
プロセッサｉからプロセッサｍｏｄ（ｉ，＃ｐｅ）＋１に
データを転送（Ｗ２からＷ１への転送）
Ｗ２のデータを使って対応する行列の更新を
各プロセッサで並列に行う
S36
ｃｔ＝ｃｔ＋１
S37
no
ｃｔ＞ｎ
S38
yes
終　了

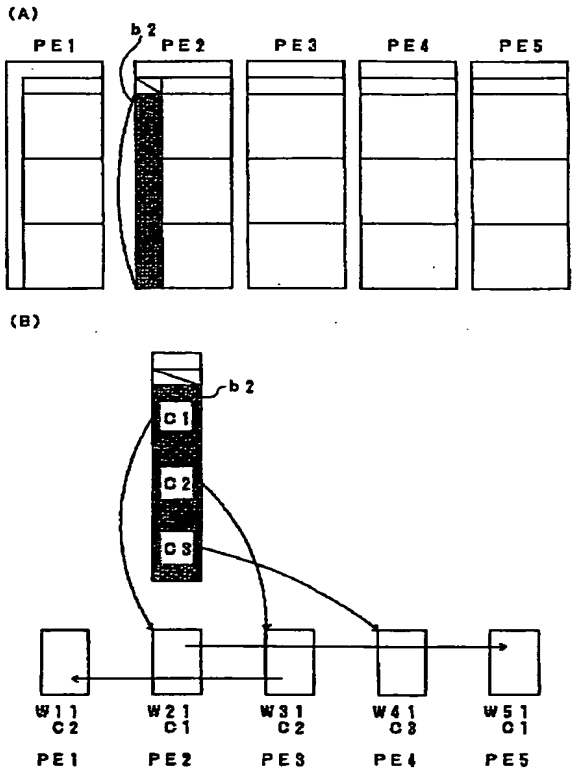
【図１５】


(A)
PE1
PE2
PE3
PE4
PE5
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
(B)
1 6 11 16
2 7 12 17
3 8 13 18
4 9 14 19
5 10 15 20

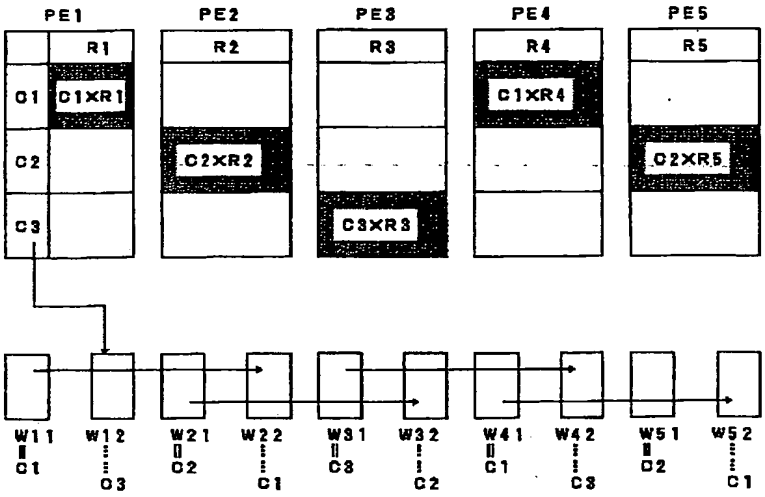
【図１６】


ブロック
b1
C1
C2
C3
W11
PE1
W21
PE2
W31
PE3
W41
PE4
W51
PE5

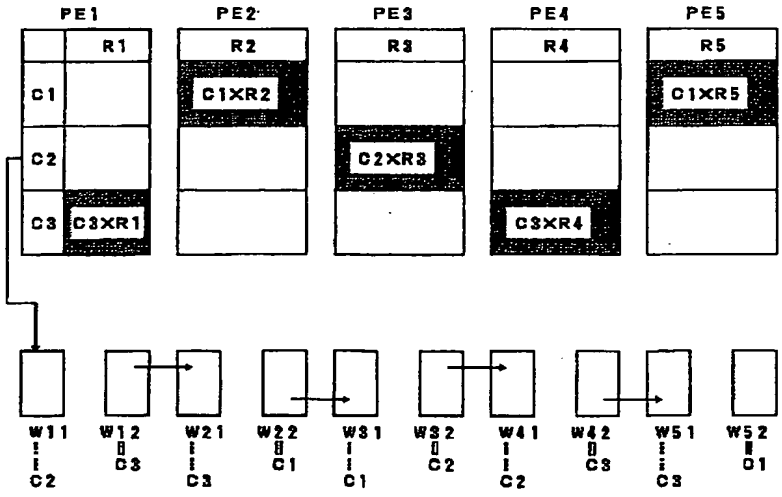
【図２０】


(A)
b2
PE1
PE2
PE3
PE4
PE5
(B)
b2
C1
C2
C3
W11
C2
PE1
W21
C1
PE2
W31
C2
PE3
W41
C3
PE4
W51
C1
PE5

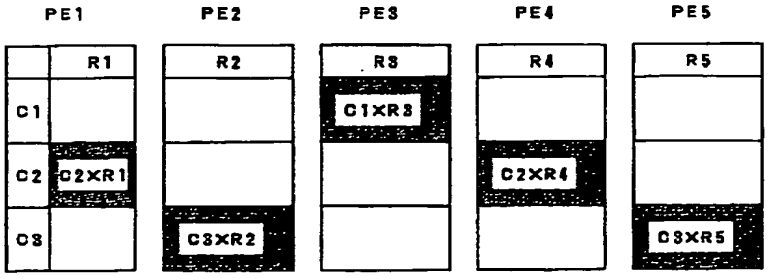
【図１７】


PE1
PE2
PE3
PE4
PE5
R1
R2
R3
R4
R5
C1
C2
C3
C1×R1
C2×R2
C3×R3
C1×R4
C2×R5
W11
C1
W12
C3
W21
C2
W22
C1
W31
C3
W32
C2
W41
C1
W42
C3
W51
C2
W52
C1

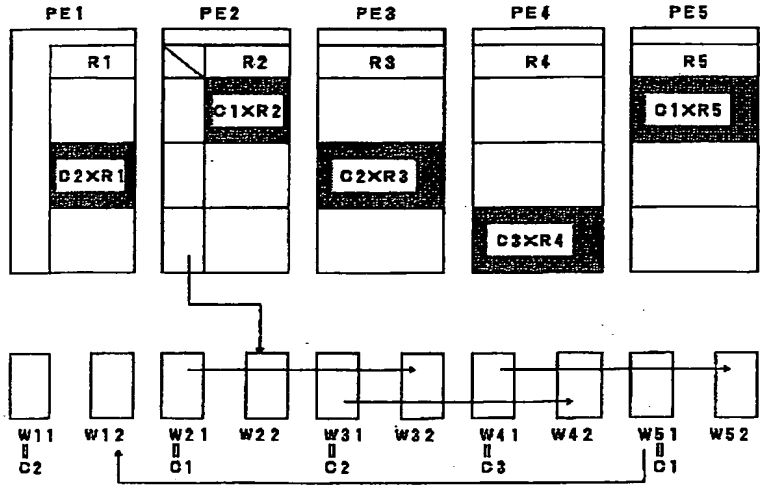
【図１８】


PE1
PE2
PE3
PE4
PE5
R1
R2
R3
R4
R5
C1
C2
C3
C3×R1
C1×R2
C2×R3
C3×R4
C1×R5
W11
C2
W12
C3
W21
C3
W22
C1
W31
C1
W32
C2
W41
C2
W42
C3
W51
C3
W52
C1

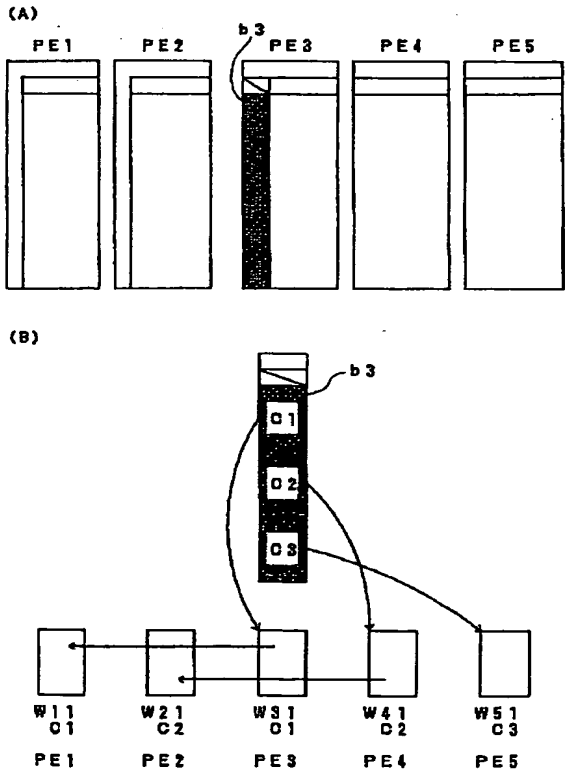
【図１９】


PE1
PE2
PE3
PE4
PE5
R1
R2
R3
R4
R5
C1
C2
C3
C2×R1
C3×R2
C1×R3
C2×R4
C3×R5

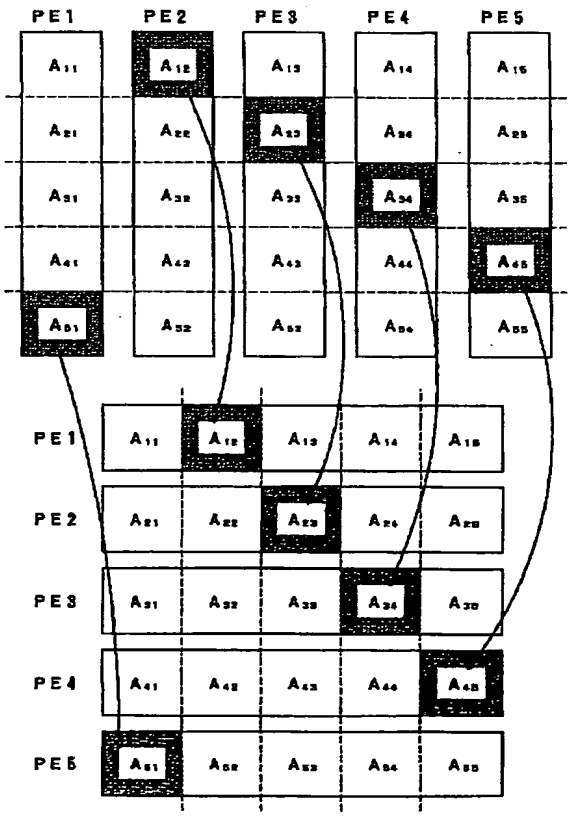
【図２１】


PE1
PE2
PE3
PE4
PE5
R1
R2
R3
R4
R5
C1×R2
C2×R1
C2×R3
C3×R4
C1×R5
W11
C2
W12
W21
C1
W22
W31
C2
W32
W41
C3
W42
W51
C1
W52

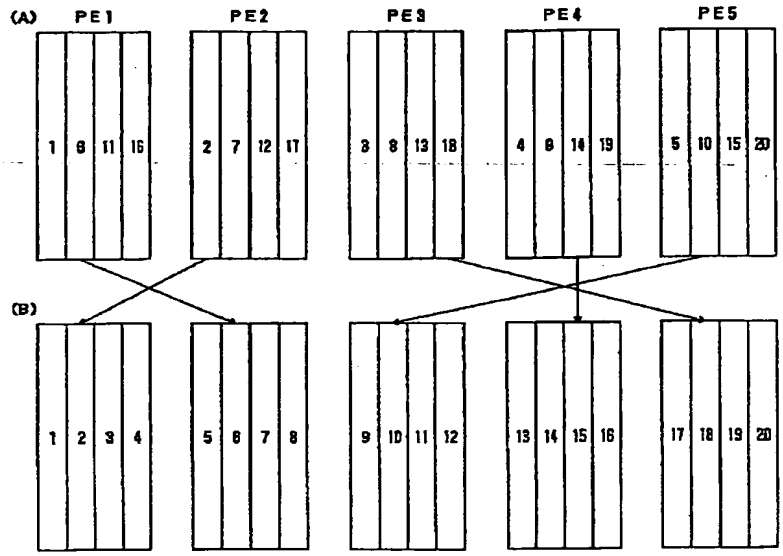
【図２２】


(A)
PE1
PE2
b3
PE3
PE4
PE5
(B)
b3
C1
C2
C3
W11
C1
W21
C2
W31
C1
W41
C2
W51
C3
PE1
PE2
PE3
PE4
PE5

【図２４】


PE1
PE2
PE3
PE4
PE5
A11
A12
A13
A14
A15
A21
A22
A23
A24
A25
A31
A32
A33
A34
A35
A41
A42
A43
A44
A45
A51
A52
A53
A54
A55

【図２３】


(A)
PE1
PE2
PE3
PE4
PE5
1 6 11 16
2 7 12 17
3 8 13 18
4 9 14 19
5 10 15 20
(B)
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20